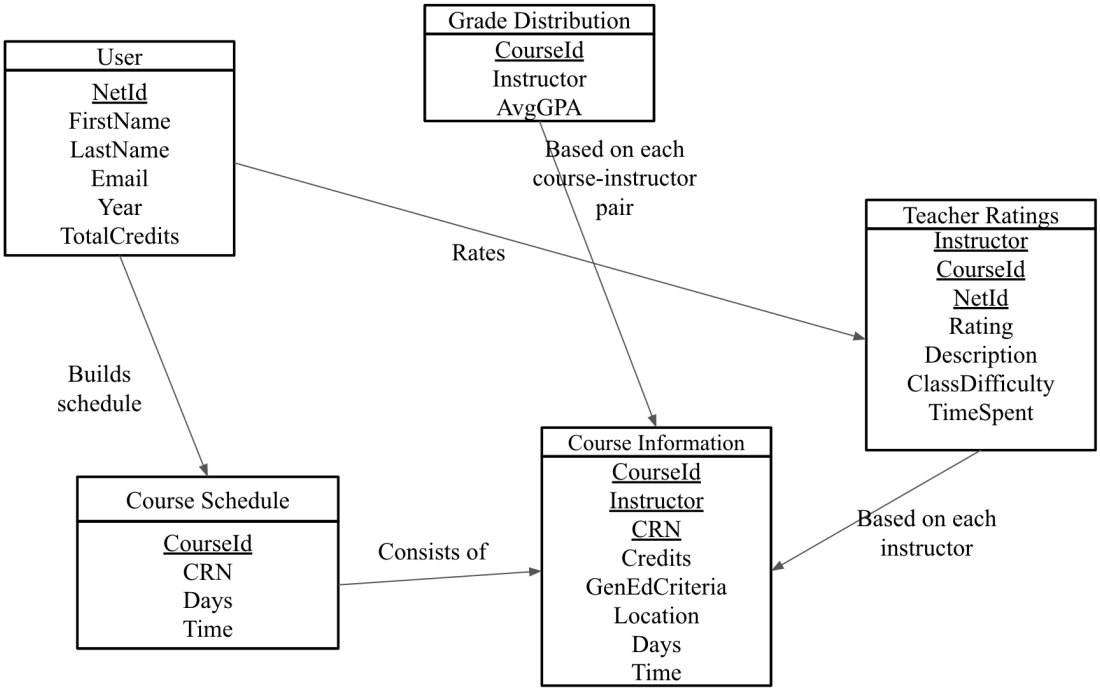
UML Diagram:



Assumptions for Entities:

User:
Each user would be a student who possesses a school NetId and is uniquely defined by this NetId. The user would also create a schedule that links to other entities like course information, teacher ratings, and grade distribution. They can review the information from those three other entities to make their scheduling decisions, as well as add in their own teacher ratings.

Course Schedule:
A course schedule is created by the user during course registration. Each course in the course schedule is uniquely defined by the CourseId. The only attributes for a schedule would be the CourseId, CRN, days, time.

Course Information:
Course information has two attributes that make it unique which are: CourseId, CRN, and Instructor. The course information reveals information pertaining to each specific course, including based on day, time, location, number of credits, and any general education it satisfies if it exists.

Grade Distribution:

Provides information about the distribution of grades for a specific course. It's uniquely identified by CourseId and it includes details such as the instructor and average GPA.

TeacherRatings:
Teacher ratings are created to allow users to add ratings for instructors and courses. The Instructor, CourseId, and NetId are primary keys which means that a unique rating entry in the TeacherRatings entity is identified by a specific Instructor, CourseId, and NetId combination. This assumes that an instructor can be rated for multiple courses, and a course can have multiple ratings from different instructors.

Assumptions for Relations:

User-Course_Schedule:
- Each user can create an account with their information
- Each user can build a schedule for themselves based on the information they review
- This is a 1-1 relationship between User and Course Schedule.

User-Teacher_Ratings:
- The teacher ratings help the user make their schedule decisions.
- Each user can view multiple teacher ratings, as well as add their own teacher ratings, so this relationship is one to many between User and Teacher Ratings.

Course_Schedule-Course_Information:
- The course schedule is built using entries from the Course Information entity.
- The user will arrange their courses based on the days and times of each course.
- Each course schedule will contain multiple courses, so this relationship between Course Schedule and Course Information is one to many.

Grade_Distribution-Course_Information:
- Each grade distribution consists of a course, teacher, and average GPA.
- This data helps the user decide which instructors are best for each course they want to schedule.
- Since each grade distribution corresponds to exactly one pair of course and instructor, this relationship is one to one.

Teacher_Ratings-Course_Information:
- Each instructor within Course Information can be rated by students within the Teacher Rating entity.
- Each instructor will have multiple ratings for different courses they have taught, but the relationship between Teacher Ratings and Course Information remains one to one.

Relational Schema:

User(
NetId: VARCHAR(10) [PK],
FirstName: VARCHAR(255),
LastName: VARCHAR(255),
Email: VARCHAR(255),
Year: VARCHAR(255),
TotalCredits: INT
)

Course_Information(
CourseId:VARCHAR(10) [PK],
Instructor: VARCHAR(255) [PK],
CRN: INT [PK],
Credits: INT,
GenEdCriteria: VARCHAR(255),
Location: VARCHAR(255)
Days: VARCHAR(255),
Time: VARCHAR(255)
)

Course_Schedule(
CourseId:VARCHAR(10) [PK, FK to Course_Information.CourseId],
CRN:INT [FK to Course_Information.CourseId],
Days: VARCHAR(255) [FK to Course_Information.Days],
Time: VARCHAR(255) [FK to Course_Information.Time]
)

Grade_Distribution(
CourseId:VARCHAR(10) [PK, FK to Course_Information.CourseId],
Instructor: VARCHAR(255),
AvgGPA: DECIMAL
)

Teacher_Ratings(
Instructor: VARCHAR(255)[PK],
CourseId: VARCHAR(10) [PK],

NetId: VARCHAR(10) [PK],
Rating: INT,
Description: VARCHAR(255),
ClassDifficulty: INT,
TimeSpent: VARCHAR(255)
)

Database Normalization:

In the User entity, regard:
NetId as A
FirstName as B
LastName as C
Email as D
Year as E
TotalCredits as F

In the Grade Distribution entity, regard:
CourseId as A’
Instructor as B’
AvgGPA as C’

In the Teacher Ratings entity, regard:
NetId as A
CourseId as A’
Instructor as B’
Rating as C’’
Description as D’’
ClassDifficulty as E’’
TimeSpent as F’’

In the Course Schedule entity, regard:
CourseId as A’
CRN as A’’’
Days as B’’’
Time as D’’’

In the Course Information entity, regard:
CourseId as A’
Instructor as B’

CRN A’’’
Credits as A’’’’
GenEdCriteria as D’’’’
Location as E’’’’
Days as B’’’
Time as D’’’

Functional Dependencies:

User: A→B,C,D,E,F (NetId → FirstName, LastName, Email, Year, TotalCredits)
Grade Distribution: A’B’→C’ (CourseId, Instructor → AvgGPA)
Teacher Ratings: A’B’A→ C’’, D’’, E’’, F’’ (NetId, CourseId, Instructor → Rating, Description, ClassDifficulty, TimeSpent)
Course Schedule; A’→ A’’’,B’’’,D’’’ (CourseId → CRN, Days, Time)
Course Information: A’B’A’’’→ A’’’’, D’’’’, E’’’’, B’’’, D’’’ (CourseId, Instructor, CRN → Credits, GenEdCriteria, Location, Days, Time)

Minimal Basis:

A→B, A→C, A→D, A→E, A→F
A’B’→C’
A’B’A→C’’, A’B’A→D’’, A’B’A→E’’, A’B’A→F’’
A’ → A’’’, A’ → B’’’, A’ → D’’’
A’→A’’’’, A’→D’’’’, A’B’A’’’→E’’’’, A’B’A’’’ → D’’’

We chose to use 3NF decomposition because we don’t have any transitive functional dependencies in our tables. The attributes in our tables are dependent on the primary keys in their respective tables. Looking at the functional dependencies of our entities, we satisfy the requirements of 3NF decomposition because we have all our keys to the left hand side in our functional dependencies and we are unable to access primary keys’ information from non-key attributes.

Changes:

We changed our initial project proposal by adding a data visualization component to display our grade visualization data. Originally we were planning to just display an average GPA, but it would be more helpful to the user to be able to compare those averages against each instructor for a certain course. When the user wants to look up grade distributions, they will look it up by course (as opposed to by course and instructor) and then be able to see all the visualizations for

that course. These will be color-coded bar graphs. This way, this grade visualization is also query-dependent.